Loops and Recursion
CSE 5211 Analysis of Algorithms
Spring 2017 (January 15, 2017)

Problem Set 2: Due Friday of Week 3

1. The pseudo-code below implements counting sort on an array \( A \) of \( n \) integers that lie in a range \([0, u)\) for some upper bound \( u \). It counts the number times each value from 0 to \( u - 1 \) occurs. It then refills the array \( A \) with that number of each value.

\[
\begin{align*}
\text{void } \text{countingSort} ( \text{array } A, \ \text{int } n, \ \text{int } u) \{ \\
\ \ \ \text{int } \text{counts}[u]; \\
\ \ \ \text{for } (\text{int } i = 0; i < u; i++) \{ \text{counts}[i] = 0; \} \\
\ \ \ \text{for } (\text{int } i = 0; i < n; i++) \{ \text{counts}[A[i]] = \text{counts}[A[i]] + 1; \} \\
\ \ \ \text{int } \text{index} = 0 \\
\ \ \ \text{for } (\text{int } i = 0; i < u; i++) \{ \\
\ \ \ \ \ \text{for } (\text{int } j = 0; j < \text{counts}[i]; j++) \{ \\
\ \ \ \ \ \ \text{A[index]} = i \\
\ \ \ \ \ \ \text{index} = \text{index} + 1 \\
\ \ \ \ \ \} \\
\ \ \ \} \\
\}
\end{align*}
\]

(a) Trace the algorithm on the array \( A \) of length \( n = 10 \) where \( u = 5 \).

\[
A = \begin{bmatrix}
2 & 4 & 3 & 2 & 4 & 3 & 0 & 1 & 0 & 4
\end{bmatrix}
\]

That is, show values in the \text{counts} array after line 4. And, show the values in the \( A \) array each time the loop at line 7 completes.

(b) What is the run time of the algorithm?

2. The power function computes \( x^n \), where \( x \) is a floating point number and \( n \) is a natural number.

(a) What is the run time complexity of this brute-force algorithm?

\[
\begin{align*}
\text{double } \text{power}(\text{double } x, \ \text{unsigned } n) \{ \\
\ \ \ \text{double } \text{product} = 1.0; \\
\ \ \ \text{for } (\text{int } i = 0; i < n; i++) \{ \\
\ \ \ \ \ \text{product} = \text{product} \times x; \\
\ \ \ \} \\
\ \ \ \text{return } \text{product}; \\
\}
\end{align*}
\]
(b) Another algorithm computes \( x^n \) by taking advantage of the identity

\[
x^n = \begin{cases} 
(x \times x)^{n/2} & \text{if } n \text{ is even} \\
(x \times x)^{\lfloor n/2 \rfloor} & \text{if } n \text{ is odd}
\end{cases}
\]

What is the worst case time complexity of the algorithm below?

```haskell
power :: Num a => a -> Int -> a
power _ 0 = 1
power x 1 = x
power x n
  | n `rem` 2 == 0 = y
  | otherwise = x * y
where y = power (x*x) (n `div` 2)
```